Michał KARBOWAŃCZYK

Technical University of Łódź

Link-level, congestion avoiding flow control solution for IP networks

Abstract. *In this paper a complete flow control solution for IP networks is presented. The mechanism consists of feedback loop and quantity-based control algorithm that utilises the Smith predictor and dead-beat control. In contrast to several proposals where flow-level control using bottleneck node approach is performed, in our solution the feedback loop is established between two neighbour nodes, what is called link-level control. The results of experiments performed in real IP network show that it is possible to completely avoid node congestion and increase throughput utilisation in the network area where the proposed solution is applied.*

Streszczenie. *W artykule przedstawiono kompletny mechanizm sterowania przepływem danych w sieciach IP. Mechanizm ten składa się z pętli zwrotnej oraz algorytmu sterowania wykorzystującego predyktor Smitha i regulator typu dead-beat.Zastosowano sterowanie na poziomie łącz, w którym pętla zwrotna jest zestawiona pomiędzy sąsiednimi węzłami sieci. Mechanizm został poddany weryfikacji eksperymentalnej w rzeczywistej sieci IP, zaś uzyskane wyniki potwierdziły jego skuteczność w zakresie eliminowania przeciążeń sieci i poprawy efektywności wykorzystania jej zasobów (S**terowanie przepływem danych w sieciach IP zapobiegające przeciążeniom**).*

Keywords: IP networks, congestion control, queue discipline
Słowa kluczowe: Sieci IP, kontrola przeciążeń, dyscypliny kolejkowania

1. Introduction

There is no manner of doubt that the demand for fast and reliable data transmission in Internet still grows. As it is shown in Cisco report [1] the total amount of data transmitted in global Internet was 31% higher in 2010 than in 2009. Furthermore, it is presented in [2] that the global demand for data transmission will quadruple by 2014, and 57% of this amount will be consumed by video streaming. It is easy to conclude that to meet such a demand, new network physical technologies, characterised by large bandwidth-delay product (BDP), must be developed. Moreover, the transmission capabilities of the network will be shared by dynamically changing number of data flows. Taking this into account it can be obviously seen that to provide desirable quality of service for those flows, proper flow control mechanisms have to be employed.

One of the most important aspects of data flow control in packet-switching networks, such as the Internet, is a congestion avoidance. Consider a network node that works in store&forward mode (i.e. Internet router). Every packet received by this node is stored in a memory buffer, where it waits until it is sent through a proper output link. If the available throughput of the output link is lower than the rate that the packets are received at, the queue of packets stored in buffer is extended. By the term of congestion we mean a situation when the memory buffer gets filled up, and subsequently received packets cannot be stored, so they must be dropped.

Dropping of packets involves several unfavourable consequences, specifically if considered flow guarantees integrity of the data being transmitted. In such case every dropped packet must be transmitted again (retransmitted). As a trivial consequence, the network capabilities are inefficiently utilised. Moreover, if some parallel flows are forced to perform retransmissions, the congested node receives more and more packets and the problem escalates. Such a phenomenon, called congestion collapse, was observed in the very early stage of Internet development [3]. Since then, a number of solutions were introduced to manage the problem of congestions in Internet. A thorough review and discussion of the most significant ones is presented in [4].

First of all, the flow control algorithm used in the Transmission Control Protocol (TCP) was extended by so called congestion window (CWND). The value of CWND determines amount of data that TCP sender is allowed to send without waiting for an acknowledgement to be returned by the TCP receiver. By decreasing or increasing this value the sender can adjust its average rate to the actual capabilities of the network. The classical AIMD (Additive Increase, Multiplicative Decrease) algorithm proposed in [5] is based on returning acknowledgements. Once every acknowledgement is received, it is assumed that the network is underloaded, and CWND is increased additively. If an acknowledgement is not returned by the receiver until so called retransmission timeout (RTO), it is assumed that the network is congested and then CWND is decreased multiplicatively. There is a significant number of features introduced to this algorithm, including selective acknowledgement (SACK) [6], fast retransmission and fast recovery [7], scalability [8 - 10] and others. Furthermore, alternative strategies of determining CWND value have been proposed. An interesting survey on these efforts is presented in [11]. Nevertheless, it must be noticed that these various TCP congestion window algorithms are common in that they do not avoid congestions in the meaning that is recognised in this paper, as CWND reduction is performed after a congestion is detected.

The operation of TCP congestion window algorithm is often supported by active queue management (AQM) schemes that are applied to memory buffer of network nodes. These algorithms selectively drop packets before the buffer gets filled up, so that the consequences of the congestion are minimised. Thus we do have to state again that they do not avoid congestions in the meaning that is recognised in this paper. Several number of AQM schemes are presented in [4] and [12].

Summarising we can clearly state that the problem of congestion avoidance on the Internet is still open. In this paper a complete solution for data flow control is presented. It utilises methods of control theory, specifically the Smith predictor and dead-beat control, to provide control algorithm that guarantees congestion avoidance and significantly increases link throughput utilisation. The proposed solution includes also a mechanism that establishes feedback loop between neighbour nodes of the network, so that the flow control can be effectively performed with no need to modify the Internet Protocol as well as transport and higher layers protocols. Furthermore, the solution can be applied to the selected network area.

The remainder of this paper is organised as follows. In the second section we present basic assumptions that are imposed on the projected flow control solution. The rationale for choosing the network layer for flow control to be applied is also discussed in this section. The third part of paper describes design and implementation of feedback

loop mechanism and control algorithm. Then results of experiments performed in a real IP network are presented and discussed. Finally the fifth section concludes the paper.

2. Basic assumptions

The overall goal of the works described in this paper is to provide a complete traffic control solution for IP networks that would completely eliminate the problem of node congestions and their negative consequences. As a result, we expect that the effective throughput (also known as "good throughput") of the network, in which our solution is applied, would be noticeably increased. Another important assumption is that it must be possible to deploy the complete traffic control mechanism in existing IP network with no necessity to modify neither the existing protocols nor network-based applications. Finally, it is required that the proposed solution can be applied to arbitrarily chosen network area.

To meet the demands mentioned above, it is necessary to carefully chose the network layer (in the terms of Internet layered network model) at which the traffic control is to be performed. Applying the solution to the transport or above layers (i.e. introducing a new transport layer protocol) would require the existing applications to be adapted to properly utilise the proposed protocol. Moreover, such a solution would be effective only for the applications that utilise the considered protocol. This problem can be easily observed if one considers parallel TCP and UDP transmissions. It is enough to establish an UDP transmission to supersede (in terms of available throughput sharing) existing TCP connections and induce network congestion. This well known problem [4, 13] is a result of lack of traffic control mechanisms in UDP and it concerns every transport layer protocol. On the other hand, the effective scope of operation for the traffic control mechanism implemented in data link layer network would be limited to the homogeneous physical network segment. Taking into account data transfer among the hosts connected by the Internet, it is clear that there are several physical segments, typically built using different data link and physical layers, that the data has to pass through. Finally, consider the network layer. Contrary to the link-level and physical layers, the network layer is unified Internet-wide. As a consequence, any solution applied at this layer may be potentially effective in world-wide scale. Furthermore, packets delivered by the network layer encapsulate data originated from all the transport layer protocols and all the applications. Recapitulating these advantages and the issues mentioned above, we decided to chose the Internet Protocol, the network layer of the Internet, as a field of operation of the traffic control solution proposed in this paper.

3. Design and implementation

As the aim of applying presented traffic control solution is to guarantee congestion avoidance, the node which originates data flow must be provided with explicit information from the network on its desired behaviour. To achieve this goal, a feedback loop, delivering control values calculated by network nodes to the source, has to be established. Two general approaches are possible in a field of designing feedback loops for telecommunication networks. In a well known flow-level control approach the loop is established between source and destination of the flow. The source prepares a dedicated data unit called control unit and sends it to the destination, which in turn sends it back. A commonly used in research works [14 - 21] variant of this approach is a bottleneck scenario. In that case one of the intermediate nodes is selected and pointed at as a bottleneck node, which is responsible for calculating the control value that determines the source behaviour. The term "bottleneck" refers to the assumption that the selected node is characterized by the worst transmission capabilities over all considered configuration.

There are two main issues concerned with the concept of flow-level control form the perspective of our work. First, it is assumed that the control values are delivered to the source of the flow, which must be aware of the control strategy. Moreover, flow-level control concept does not fully conform to the specificity of packet-switching networks. In such a network, even if we consider a flow controlled by a connection-oriented transport layer protocol (i.e. TCP), the route from the source to the destination may be different for every single data unit (packet) transmitted.

In the less common link-level control approach the control system consists of two neighbour nodes[1]. Considering an example of a single, unidirectional data flow that passes three neighbour nodes P, R and N, depicted in Fig. 1, two pairs of neighbour nodes can be distinguished in this configuration: P-R and R-N.

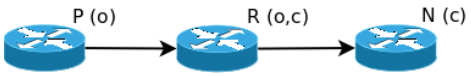

Fig. 1. A simple configuration of neighbour nodes

The two pairs of nodes correspond with two configurations of link-level control: one with node R as a controller (c) and P as a controlled object (o), and one with node N as a controller and R as a controlled object.

A set of pairs of neighbour nodes forms a route that the packet passes through being delivered from the source to the destination. Even if such a route is subject to change for the considered data flow, these changes do not affect the way that the particular pairs of neighbour nodes operate. Moreover, it is possible that the flow control is performed only for selected pairs of neighbour nodes, what fulfils the requirement that the proposed solution can be applied to arbitrarily chosen network area.

Since the specification of the Internet Protocol does not provide any facilities that could be utilised to establish closed loops between neighbour nodes of IP network, it was necessary to extend the functionality of IP node (router). First we state that the considered node may operate as a controller in multiple link-level control configurations, as there may be multiple elementary flows that pass it through. Thus, a node that acts as a controller must be able to determine a list of such flows. This problem is already solved in Linux operating system kernel by connection tracking module (*conntrack*), which is a core part of Linux network stack. We extended this module and also created a new packet filter that reads and stores source data link layer addresses of incoming frames. As a result, the extended connection tracking module that operates on considered network node is able to provide a list of its predecessor nodes (namely, a list of their data link layer addresses) along with a list of elementary flows that are originated by (in terms of link-level flow control) every such a node.

The proposed flow control solution utilises a discrete time control algorithm presented in details in [22]. It is run by the controller at time instants kT, where T denotes the discretisation period and k numbers subsequent cycles. The determined control value designates amount of data that

[1] We regard two nodes as neighbour if they are connected to each other without intermediation of any nodes at the same network layer

the controlled object is allowed to send. It is calculated with following formula

(1) $$a(k) = X^D - x(k) - w(k)$$

where: $a(k)$ denotes a control value, $x(k)$ represents the length of the queue stored by node in its buffer at time instant kT, and $w(k)$ is the sum of control values that were already sent to the controlled nodes, but they have not affected the state of the controller node yet (the controlled nodes have not so far received them, or they have not enough amount of data to be sent). X^D is a reference value chosen by network administrator. The distribution of the control value obtained from (1) among neighbour nodes is proportional to the number of elementary flows originated by each neighbour node. The proposed algorithm is characterised by several favourable properties described in form of mathematical theorems and strictly proven in [22]: it calculates nonnegative and bounded control values, guarantees that the length of the packet queue in memory buffer of controller node never exceeds reference value X^D., and finally, if the reference value and consequently the capacity of memory buffer meet the minimal condition formulated in [22], it guarantees effective utilisation of available throughput. The control algorithm has been implemented in a form of queue discipline, that is, a facility of Linux network stack, responsible for managing the queue of the packets awaiting to be sent through a specified network interface.

4. Experimental results

We examined the proposed solution through a number of experiments that were run in a real IP network. Due to considerable extensiveness of the results and analysis, we decided to select three experiments to be presented in this paper; a broader range of them can be found in [22].

Experiment 1

The goal of the experiment was to determine how operation of the proposed solution affects the way in which several TCP connections share variable, unpredictable link throughput. Every node of the considered network was running Fedora operating system based on Linux [24] kernel version 2.6.26, extended according to the description provided in previous sections. The experimental results were obtained from two sources. First, we utilised *tcpdump* network sniffer [25] to record the network traffic on considered nodes. These records were processed using *wireshark* traffic analyser [26] to obtain the transmission rates of considered flows, calculate amount of transmitted data and detect data retransmissions. Moreover, the proposed queuing discipline was set to record its log, regardless of the control algorithm was in operation or not. Such a log contains, among others, information on queue length $x(t)$, packet queuing delay $o(t)$, amount of data that was lost since its last run $l(t)$ (the unit of this value is B/T - bytes per period) and total amount of data that was lost during the experiment.

To perform the tests we set up a topology presented in Fig. 2.

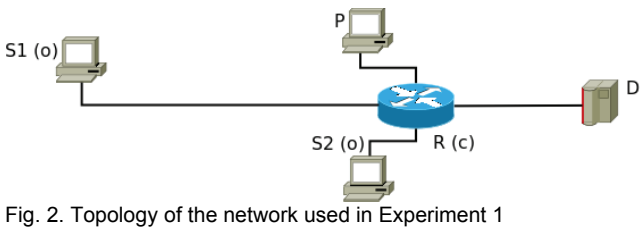

Fig. 2. Topology of the network used in Experiment 1

The node R is an IP router. Every link connecting the nodes was an isolated 10 Mb/s Ethernet segment. The proposed queuing discipline is deployed in nodes S1, S2, and R. Round trip time RTT of every circuit consisting of neighbour nodes was below 1 ms except the pair R-D, where it was 50 ms. We assumed that the time scheduling period T is 10 ms, however from the obtained results we concluded that it was in fact realised with +/- 1 ms irregularities. Taking this into account, to calculate the reference value according to the condition formulated for full utilisation of available throughput, we assumed that T = 11 ms and finally obtained X^D = 30 kB.

At $t \approx 1$ s the source S1 establishes a TCP connection with the destination D. So does the source S2 at $t \approx 2$ s. Both connections are fed with data so that they are able to emit packets continuously at 10 Mb/s. At $t \approx 4$ s the node P starts sending data to D. It utilises UDP datagrams and does not perform any form of flow control. The configuration of R provides that the packets received from P take precedence over those received from S1 and S2. This causes processing of TCP flows by R to be practically suspended during the activity of P. This scenario was run in two variants. In variant 1 (v. 1) queuing disciplines in nodes S1, S2 and R do not utilise proposed control algorithm, so there is no additional (beyond the standard TCP congestion window mechanism) flow control applied to any node. In variant 2 (v. 2) proposed flow control solution operates in configurations S1-R and S2-R.

The results of the experiment are shown in figures below. Fig. 3. depicts transmission rates, and Fig. 4. presents queue length and data loss in the router R.

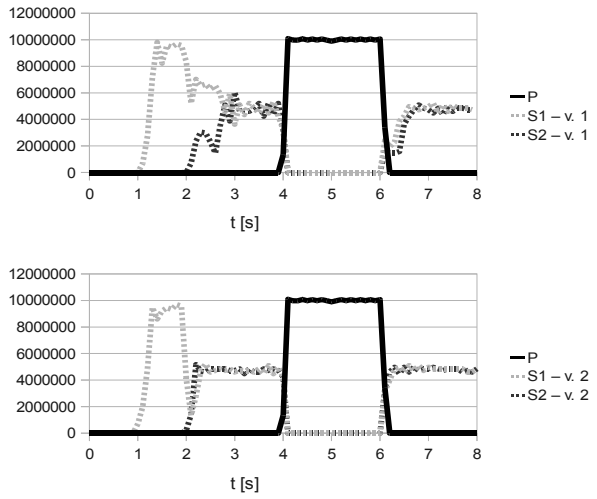

Fig. 3. Transmission rates [b/s], Experiment 1, node R

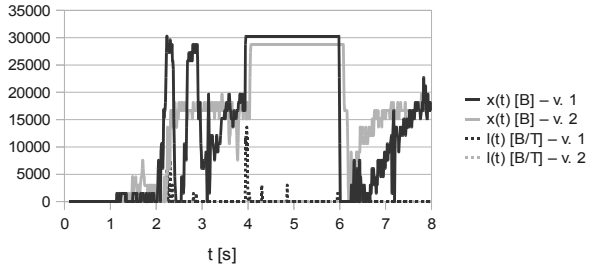

Fig. 4. Queue state, Experiment 1, node R

First let focus on the transmission rates of TCP connections between 2nd and 4th second of experiment. As TCP flow from S2 begins to transmit data, the flow from S1 is forced to reduce its congestion window and emission

rate. In both variants the TCP connections tend to equally share the throughput of link between R and D, however, in case of variant 1 we observe oscillations of transmission rates and perceptible amount of data lost due to congestion of node R. Contrary to these notices, in case of variant 2 the available throughput is shared among TCP flows more smoothly and efficiently than in case of variant 1, and data loss is eliminated. Similar results can be observed after node P finishes its transmission.

In addition to the results presented above, total amount of data transmitted by both TCP connections through router R was measured in every variant of the scenario. Subtracting the amount of data that was retransmitted, we obtain an effective transmission statistics (so called "good throughput").

Table 1. Transmission statistics obtained in Experiment 1

Quantity	Variant 1 [B]	Variant 2 [B]
S1 – total transmission	3286374	3194260
S1 – retransmissions	28766	0
S2 – total transmission	1917126	2261146
S2 – retransmissions	52990	0
S1 + S2 – total transmission	5203500	5455406
S1 + S2 – retransmissions	81756	0
S1 + S2 – effective transmission	*5121744*	*5455406*

The results presented in Table 1. can be summarized as follows. The most important factor is the amount of data that was effectively transmitted to the destination D, located in the last row. Taking as the base the result of experiment run in variant 1, one can easily see that applying the proposed solution in variant 2 caused that the good throughput was increased by 6.5%.

Experiment 2

The goal of the experiment was to determine how operation of the proposed solution affects the way in which several TCP connections and UDP transmissions share variable, unpredictable link throughput. Particularly we aimed at the influence of proposed flow control solution on UDP transmissions, as there is no flow control performed by UDP protocol itself.

The experiment was performed using network topology and scenario similar to those used in Experiment 1. The node S3 was added, as depicted in Fig. 5.

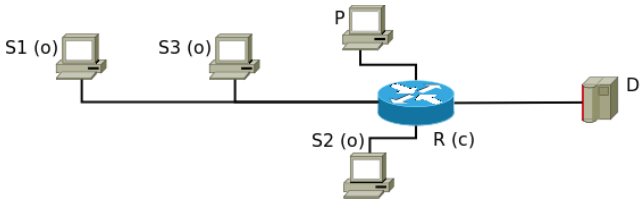

Fig. 5. Topology of the network used in Experiment 2

The source S3 starts sending UDP datagrams to D at $t \approx 3$ s. Its transmission rate is limited to about 8 Mb/s by the application that provides data to be sent.

The scenario mentioned above was run in two variants similarly to the variants of Experiment 1.

The results of the experiment are shown in figures below. Fig. 6. depicts transmission rates, and Fig. 7. presents queue length and data loss in the router R.

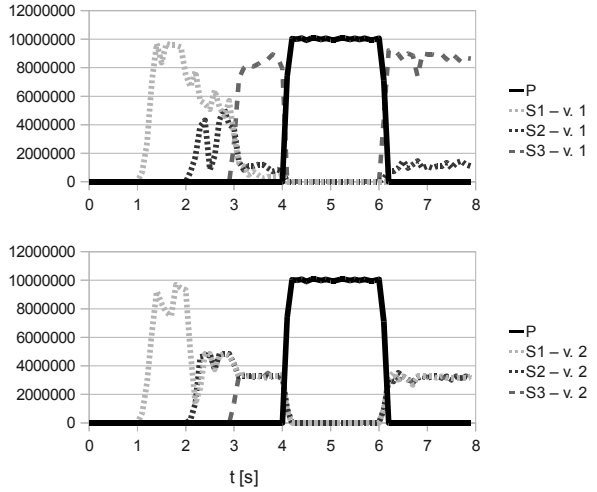

Fig. 6. Transmission rates [b/s], Experiment 2, node R

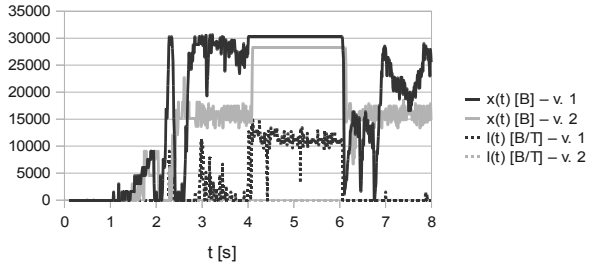

Fig. 7. Queue state, Experiment 2, node R

Note that the results obtained during the first three seconds of the experiment are similar to those acquired from Experiment 1. However, just after S3 starts sending UDP datagrams, in case of variant 1 it quickly achieves its maximum admissible (due to the limitation mentioned in experiment scenario) transmission rate. Consequently, both TCP connections are forced to reduce their transmission rates, as they share only the throughput not consumed by S3 UDP transmission. We also observe significant data loss during the activity of node P.

Both these phenomenons are eliminated in case of variant 2. The available throughput is equally distributed among the considered flows, regardless of the transport layer protocol that controls given flow. Furthermore, we observe that no data loss occurred.

Similarly to the previous experiment, total amount of data transmitted by considered flows through the router R was measured in every variant of the scenario.

Table 2. Transmission statistics obtained in Experiment 2

Quantity	Variant 1 [B]	Variant 2 [B]
S1 – total transmission	1701776	2542254
S1 – retransmissions	36336	0
S1 – total transmission	762512	1621886
S1 – retransmissions	52990	0
S1 + S2 – total transmission	2464288	4164140
S1 + S2 – retransmissions	89326	0
S1 + S2 – effective transmission	2374962	4164140
S3 – total transmission	3095134	1162688
S1 + S2 + S3 – total transmission	5559422	5326828
S1 + S2 + S3 – effective transmission	*5470096*	*5326828*

Taking into account the statistics obtained in both variants it is necessary to investigate the phenomenon that after the link-level flow control was applied, the amount of data effectively transmitted by all three considered flows decreased by about 2.5%. Nevertheless, the conclusion that the proposed flow control solution imposed a negative effect on network performance is easy to refute through following substantiation. The total amount of data lost by the router R during variant 1, recorded by the queuing discipline, is about 2 400 000 B. On the other hand, the total amount of data retransmitted by TCP connections is less than 100 000 B. This leads to the conclusion that at least about 2 300 000 B of lost data belongs to the UDP transmission. Comparing this to the total amount of data that was successfully transmitted by S3, we state that about 40% of data emitted by this node is lost at the router R due to its congestion. From the practical perspective it is very probable that whole UDP transmission is then just useless, and only the TCP flows should be counted in good throughput in case of variant 1.

Experiment 3

The goal of the experiment was to examine the behaviour of the proposed solution in situation when one of the controlled nodes does not send as much data as it is allowed according to the received control values. In practice such a situation is possible due to two typical reasons: the considered node has no data to be sent or its available throughput is not sufficient.

To perform the tests we set up a topology presented in Fig. 8.

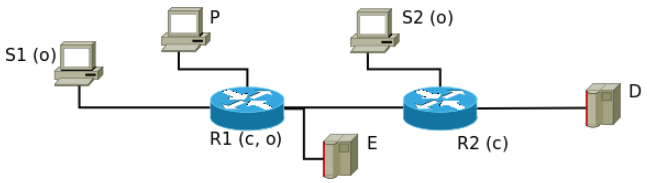

Fig. 8. Topology of the network used in Experiment 3

The nodes R1 and R2 are IP routers. At $t \approx 1$ s both sources S1 and S2 establish a TCP connections with the destination D. At $t \approx 3$ s the node P starts sending data to E. It utilises UDP datagrams and does not perform any form of flow control. The configuration of R1 provides that the packets received from P take precedence over the ones received from S1. The practical consequence is that the throughput available at R1 for the flow from S1 is significantly reduced. This scenario was run in two variants similarly to the previous experiments; in variant 2 proposed flow control solution operates in configurations S1-R1, R1-R2 and S2-R2.

The results of the experiment are shown in figures below. Fig. 9. depicts transmission rates measured at nodes R1 and R2.

Note that the results obtained during the first three seconds of the experiment are similar to those acquired from Experiment 1. However, just after node P starts sending UDP datagrams, we observe that practically no data from S1 is transmitted by R2. This is obviously caused by the fact that during this period R1 does not transmit data to R2 due to the reason mentioned in the scenario of the experiment. In case of variant 1 the TCP flow from S2 utilises the supplementary available throughput and increases its transmission rate. It is noticeable that even after node P finishes its transmission, TCP flow from S2 keeps utilising more of available throughput than the flow from S1. On the other hand, in case of variant 2 the node R2 surprisingly stops transmitting data originated by S2 (however, after node P finishes its transmission, both TCP connections share the throughput available for node R2 equally). The consequences of this phenomenon can be observed by comparing total amount of data transmitted by considered flows through the router R2 measured in both variants, which is presented in Table 3.

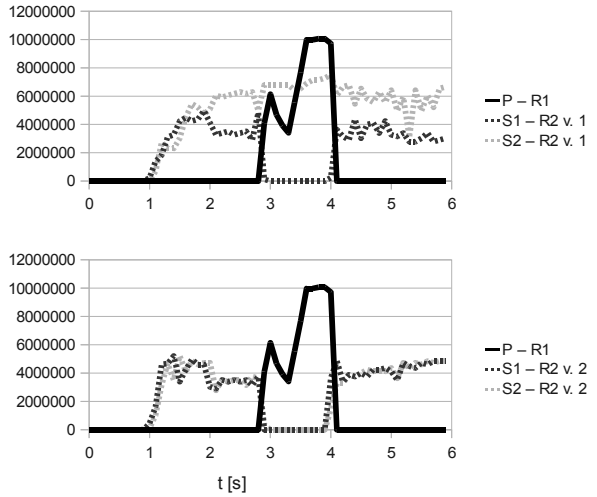

Fig. 9. Transmission rates [b/s], Experiment 3, nodes R1 and R2

Table 3. Transmission statistics obtained at R2 in Experiment 3

Quantity	Variant 1 [B]	Variant 2 [B]
S1 – total transmission	1600522	1921306
S1 – retransmissions	15140	0
S2 – total transmission	3472876	1877400
S2 – retransmissions	24224	0
S1 + S2 – total transmission	5073398	3798706
S1 + S2 – retransmissions	39364	0
S1 + S2 – effective transmission	*5034034*	*3798706*

One can easily see that applying the proposed solution in variant 2 caused that the good throughput was decreased by 24.5%; specifically, the good throughput of TCP flow from S2 was reduced almost by half.

The considered phenomenon can be explained by examining the behaviour and internal state of the queue discipline that operates at node R1. Fig. 10. depicts the transmission rate, queue length and the sum of the control values stored for further utilisation (denoted as $wQ(t)$) by this node. For the sake of clarity, only the time period of P activity is presented.

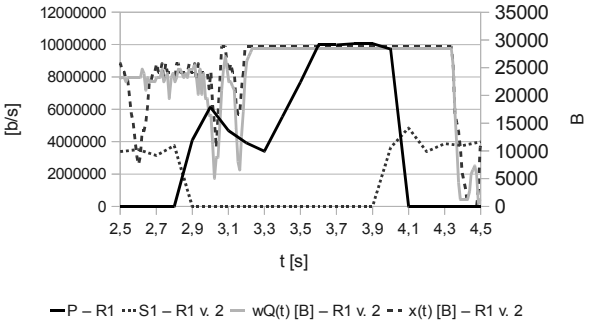

Fig. 10. Queue discipline state, Experiment 3, node R1

First notice that the buffer of the node R1 is filled up with data that can not be sent due to lack of available

throughput. However, as long as the node R2 sends its data, it generates positive control values $a(k)$ and distributes them equally among R1 and S2. The actual problem is that the node R1 accumulates the control values received from R2 up almost to the reference value X^D (as one can easily observe in Fig. 10.) increasing the factor $w(k)$ in formula (1) utilised by the control algorithm operating at R2. Consequently, at some moment the control value sent by R2 to S2 is smaller than the size of a single packet, what prevents S2 (and in turn R2) from sending any data, and the whole configuration becomes stalled until R1 starts sending the data to R2. The conclusion of Experiment 3 is that the proposed control solution should be refined so that it avoids excessive accumulation of control values by the controlled node that is not able to send the data.

5. Summary

In this paper a new flow control solution was examined. It utilises facilities well known in control theory, namely the Smith predictor and dead-beat control, to control amounts of data that the controlled nodes are obliged to send. The tests performed in a real IP network confirm that the proposed control algorithm is characterised by significant favourable properties, presented in form of mathematical analysis in [22]. It completely avoids congestion and data loss, thus increasing the efficiency of available throughput utilisation. Moreover, as the flow control is performed at network layer level, it influences every considered flow in the same manner, regardless of which transport layer protocol controls is involved. Finally, the direction of further development of the proposed control algorithm, connected with the problem of excessive accumulation of control values by the controlled node, was pointed.

The core theses of this paper were presented at XIV International Conference System, Modelling and Control SMC'2011.

REFERENCES

[1] Cisco Public Information, Cisco Visual Networking Index: Usage, 2010
[2] Cisco Public Information, Cisco Visual Networking Index: Forecast and methodology 2009–2014, 2010
[3] Nagle J., RFC 896: Congestion control in IP/TCP internetworks, 1984
[4] Welzl M., Network congestion control: managing internet traffic. *Wiley*, 2005
[5] Jacobson V., Braden R., RFC 1072: TCP extensions for long-delay paths, 1988
[6] Mathis M. et al., RFC 2018: TCP selective acknowledgment options, 1996.
[7] Allman M., Paxson V., Stevens W., RFC 2581: TCP congestion control, 1999
[8] Kelly T., Scalable TCP: improving performance in highspeed wide area networks. *Computer Communications Review*, 32 (2003), No. 2
[9] Floyd S., RFC 3649: HighSpeed TCP for large congestion windows, 2003
[10] Leith D., Shorten R., H-TCP: TCP for high-speed and long-distance networks. *Proceedings of 2nd PFLDnet workshop*, Argonne, 2004
[11] Qureshi B., Othman M., Hamid N.A.W., Progress in various TCP variants. *2nd International Conference on Computer, Control and Communication*, Karachi, 2009
[12] Chatranon G., Labrador M. A., Banerjee S., A survey of TCP-friendly router-based AQM schemes. *Computer Communications*, Volume 27 (2004), No. 15
[13] Grzech A., Sterowanie ruchem w sieciach teleinformatycznych. Oficyna Wydawnicza Politechniki Wrocławskiej, 2002
[14] Mascolo S., Congestion control in high-speed communication networks using the Smith principle, *Automatica*, 35 (1996), No. 12, 1921-1935
[15] Mascolo S., Smith's principle for congestion control in high-speed data networks, *IEEE Transactions on Automatic Control*, 45 (2000), No. 2, 358-364
[16] Mascolo S., Dead-time and feed-forward disturbance compensation for congestion control in data networks, *International Journal of Systems Science*, 34 (2003), No. 10-11, 627-639
[17] Bartoszewicz A., Molik T., ABR traffic control over multi-source single-bottleneck ATM networks, *Journal of Applied Mathematics and Computer Science*, 14 (2004), No. 1, 43-51
[18] Karbowańczyk M., Bartoszewicz A., Flow control in a single connection ATM network with a limited source capability. *Journal of Applied Computer Science*, 13 (2005)
[19] Bartoszewicz A., Nonlinear flow control strategies for connection oriented communication networks, *Proceedings of the IET Part D: Control Theory and Applications*, 153 (2006), No. 1, 21-28
[20] Gómez-Stern F., Fornés J.M., Rubio F.R., Dead-time compensation for ABR traffic control over ATM networks, *Control Engineering Practice*, 10 (2002), No. 5, 481-491
[21] Pietrabissa A., Delli Priscoli F., Fiaschetti A., Di Paolo F., A robust adaptive congestion control for communication networks with time-varying delays, *Proceedings of the IEEE International Conference on Control Applications, German,* 2006, 2093-2098
[22] Karbowańczyk M., Zapobieganie przeciążeniom w sieciach IP z zastosowaniem sprzężenia zwrotnego oraz predyktora Smitha i regulatora dead-beat, *rozprawa doktorska, Politechnika Łódzka, Instytut Informatyki*, 2011
[23] The Linux kernel project, http://www.kernel.org/
[24] Tcpdump packet analyzer, http://www.tcpdump.org/
[25] Wireshark network protocol analyzer, http://www.wireshark.org/

***Author**: dr inż. Michał Karbowańczyk, Institute of Information Technology, Technical University of Łódź, michal.karbowanczyk@p.lodz.pl*